Convolutional Neural Networks for Dialogue State Tracking without Pre-trained Word Vectors or Semantic Dictionaries

Mandy Korpusik, James Glass
MIT Computer Science and Artificial Intelligence Lab, Cambridge MA USA
{korpusik, glass}@mit.edu

1. Goal

Avoid reliance on manual feature engineering for dialogue state tracking.

- Neural models instead of rule-based.
- Spoken language understanding (SLU) and dialogue state tracking (DST) in a single model, rather than a pipeline of modules.
- No hand-crafted semantic dictionaries for delexicalizing the user query.

Slot-Value	Synonyms
Food=Cheap	[affordable, budget, low-cost, low-priced, ...]
Area=Centre	[center, downtown, central, city centre, ...]
Rating=High	[best, high-rated, highly rated, top-rated, ...]

- No pre-trained character or word vectors injected with semantic information.

2. WOZ 2.0 Task

Predict all the user's slots at each turn in a restaurant booking dialogue.

User utterances are *written*, requiring semantic understanding.

User: Is there any place here in the centre that serves corsica food? `food = corsica; area = centre`
System: What price range are you looking for? **User:** Any price range will do. `food = corsica; area = centre; price = dontcare`
System: There are no restaurants available matching your criteria. Would you like to try a different area, price range, or food type? **User:** Are there any restaurants in the centre that serves North American type of food? `food = north_american; area = centre; price = dontcare`

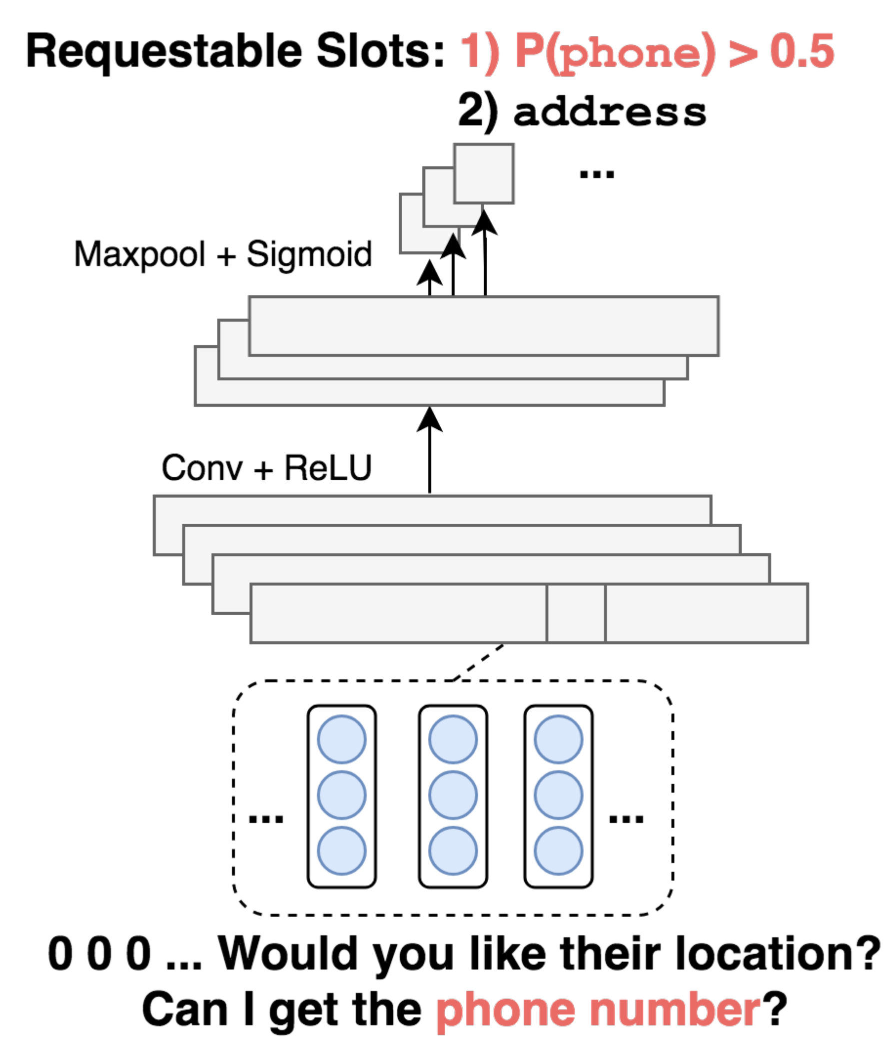
Two slot types are predicted:

- **Requestable**: user *requests* information about a restaurant (e.g., phone, address).
- **Informable**: user *informs* the system of their preference (e.g., cuisine, price).

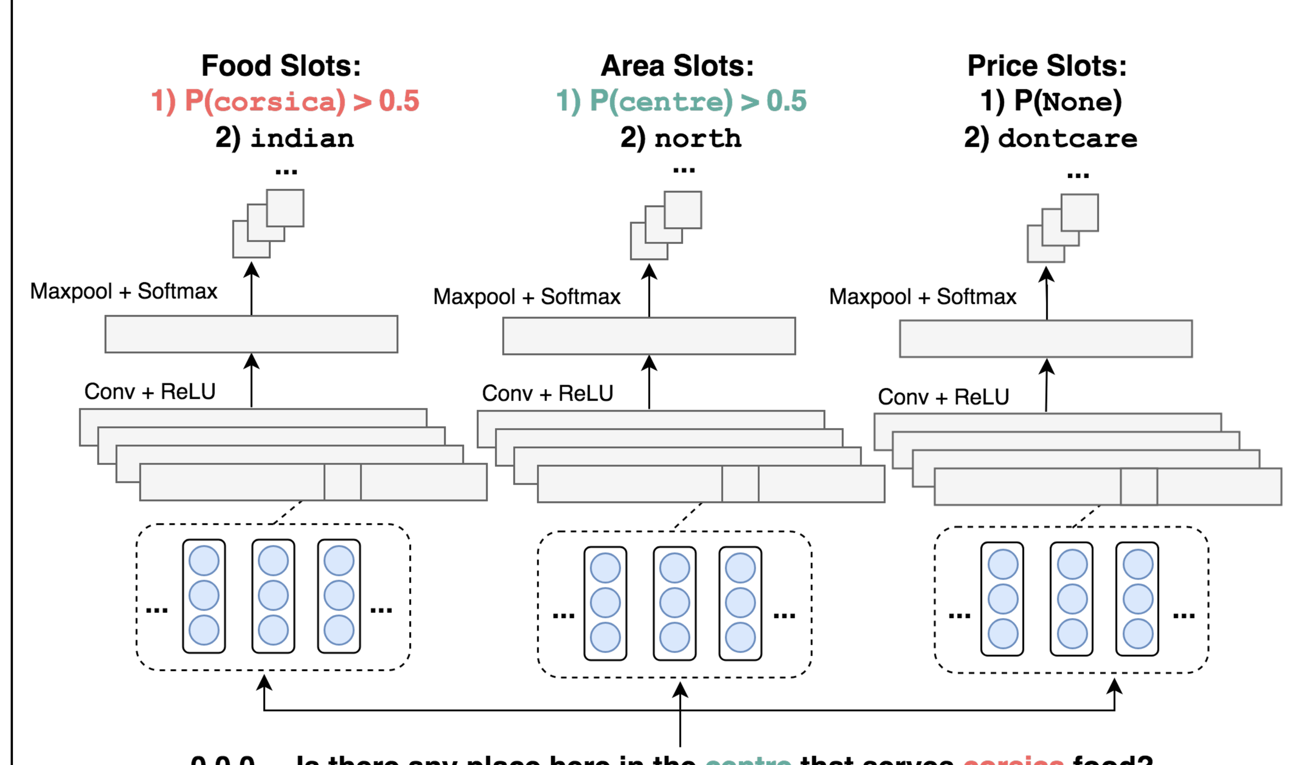
Slot	Type	Num Values
Food	Informable, Requestable	75
Area	Informable, Requestable	7
Pricerange	Informable, Requestable	4
Name	Requestable	N/A
Address	Requestable	N/A
Phone	Requestable	N/A
Postcode	Requestable	N/A
Signature	Requestable	N/A

3. Neural Models

Requestable slots model: one CNN with separate binary output layers for each requestable slot.



Informable slot models: separately trained CNN for each slot, with softmax across all values (and None).



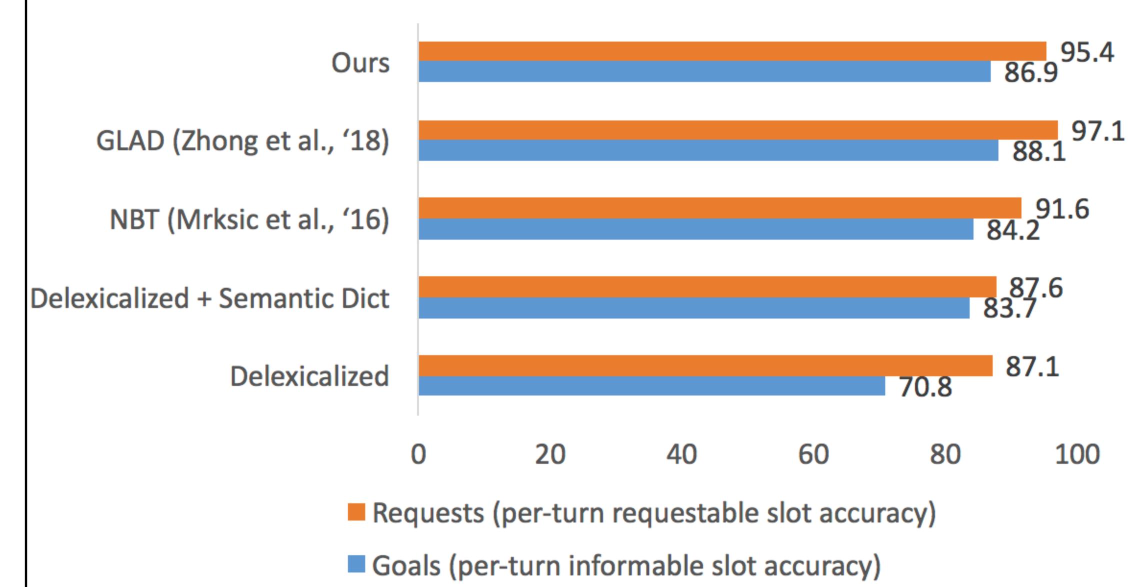
4. Post-Processing

Check for any missing informable slots:

- For slots that were requested by the system in that turn, but where the top predicted slot value was None, take the second highest slot value.
- Do string matching on the user utterance for any exact match slot values that were missed.

Tune threshold hyperparameters on the development set for adding new slots and updating existing slots.

5. Results



6. Analysis

Errors require deep semantic understanding:

User: Hello, I'm looking for a **nice** restaurant with vegetarian food. **True**: `food = vegetarian` **Pred**: `food = vegetarian; price = ` **`expensive`**
User: Hi, I want a Tuscan restaurant that's **expensively** priced. **True**: `food = tuscan; price = ` **`expensive`** **Pred**: `food = vegetarian; price = ` **`cheap`**
System: No such results found. Would you like me to search for any Mediterranean restaurants in the **centre**? **User:** Is there a Lebanese place **anywhere** around? **True**: `food = lebanese; area = ` **`dontcare`**`; price = dontcare` **Pred**: `food = lebanese; area = ` **`centre`**`; price = dontcare`
User: I like Persian but I'm close to **broke**. **True**: `food = persian; price = ` **`cheap`** **Pred**: `food = persian`
System: I will search for the most nearby English restaurant. **User:** It should be an **upscale** English restaurant. **True**: `food = english; price = ` **`expensive`** **Pred**: `food = english`

CNN filters learn to focus on different slots:

CNN Filter	Top-10 Tokens
11	caribbean, indian, type, food, bistro, serve, something, thai, singaporean, romanian
13	european, canapes, indian, bistro, japanese, caribbean, world, persian, italian, british
16	postcode, post, center, thank, restaurant, then, i, need, could, uh
19	phone, telephone, does, their, the, is, south, east, i, in
50	code, expensive, type, moderate, serving, kind, any, my, anything, cheap

7. Conclusion

CNN models without semantic dictionaries or pre-trained word vectors are **competitive with state-of-the-art**, reaching 95.4% requestable and 86.9% joint goal accuracy on WOZ 2.0.

In the future, we plan to experiment on the noisy *spoken* test set of DSTC2.